```sql
/*
Question 1
*/

-- Creating the CUSTOMER Table
CREATE TABLE CUSTOMER (
    CUSTOMER_ID INT PRIMARY KEY,
    FIRST_NAME VARCHAR(50),
    SURNAME VARCHAR(50),
    ADDRESS VARCHAR(100),
    CONTACT_NUMBER VARCHAR(20),
    EMAIL VARCHAR(100)
);

-- Creating the EMPLOYEE Table
CREATE TABLE EMPLOYEE (
    EMPLOYEE_ID VARCHAR(10) PRIMARY KEY,
    FIRST_NAME VARCHAR(50),
    SURNAME VARCHAR(50),
    CONTACT_NUMBER VARCHAR(20),
    ADDRESS VARCHAR(100),
    EMAIL VARCHAR(100)
);

-- Creating the DONATOR Table
CREATE TABLE DONATOR (
    DONATOR_ID VARCHAR(10) PRIMARY KEY,
    FIRST_NAME VARCHAR(50),
    SURNAME VARCHAR(50),
    CONTACT_NUMBER VARCHAR(20),
    EMAIL VARCHAR(100)
);

-- Creating the DONATION Table
```

```
    DONATION_ID INT PRIMARY KEY,
    DONATOR_ID VARCHAR(10),
    DONATION VARCHAR(100),
    PRICE DECIMAL(10, 2),
    DONATION_DATE DATE,
    FOREIGN KEY (DONATOR_ID) REFERENCES DONATOR(DONATOR_ID)
);

-- Creating the DELIVERY Table
CREATE TABLE DELIVERY (
    DELIVERY_ID INT PRIMARY KEY,
    DELIVERY_NOTES VARCHAR(200),
    DISPATCH_DATE DATE,
    DELIVERY_DATE DATE
);

-- Creating the RETURNS Table
CREATE TABLE RETURNS (
    RETURN_ID VARCHAR(10) PRIMARY KEY,
    RETURN_DATE DATE,
    REASON VARCHAR(200),
    CUSTOMER_ID INT,
    DONATION_ID INT,
    EMPLOYEE_ID VARCHAR(10),
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID),
    FOREIGN KEY (DONATION_ID) REFERENCES DONATION(DONATION_ID),
    FOREIGN KEY (EMPLOYEE_ID) REFERENCES EMPLOYEE(EMPLOYEE_ID)
);

-- Creating the INVOICE Table
CREATE TABLE INVOICE (
    INVOICE_NUM INT PRIMARY KEY,
    CUSTOMER_ID INT,
    INVOICE_DATE DATE,
    DONATION_ID INT,
    DELIVERY_ID INT,
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID),
    FOREIGN KEY (EMPLOYEE_ID) REFERENCES EMPLOYEE(EMPLOYEE_ID),
    FOREIGN KEY (DONATION_ID) REFERENCES DONATION(DONATION_ID),
    FOREIGN KEY (DELIVERY_ID) REFERENCES DELIVERY(DELIVERY_ID)
);


-- Populating the CUSTOMER Table using INSERT statements
INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CO
VALUES (11011, 'Jack', 'Smith', '18 Water Rd', '0877727521', 'jsmit

INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CO
VALUES (11012, 'Pat', 'Hendricks', '22 Water Rd', '0872268357', 'ph

INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CO
VALUES (11013, 'Clark', 'Sam', '15 Ocean Way', '0878724453', 'clark

INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CO
VALUES (11014, 'Kevin', 'Jones', '55 Mountain Way', '0822345556', '

INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CO
VALUES (11015, 'Lucy', 'Williams', '5 Main Rd', '0827388521', 'lw@m


-- Populating the EMPLOYEE Table using INSERT statements
INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUM
VALUES ('emp101', 'Jeff', 'Davis', '0877727521', '10 Main Rd', 'jd@

INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUM
VALUES ('emp102', 'Kevin', 'Marks', '0837737522', '18 Water Rd', 'k

INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUM
INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUM
VALUES ('emp104', 'Adebayo', 'Dryer', '0797115244', '1 Sea Road', '

INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUM
VALUES ('emp105', 'Xolani', 'Samson', '0827122255', '12 Main Road',


-- Populating the DONATOR Table using INSERT statements
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBE
VALUES ('20111', 'Jeff', 'Watson', '0827721250', 'jwatson@gmail.com

INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBE
VALUES ('20112', 'Stephen', 'Jones', '0838775602', 'sjones@gmail.co

INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBE
VALUES ('20113', 'Abraham', 'Clark', '0827655430', 'aclark@gmail.co

INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBE
VALUES ('20114', 'Kelly', 'Koe', '0823657650', 'kkoe@isat.com');

INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBE
VALUES ('20115', 'Alice', 'Klay', '0797656430', 'aklay@gmail.com');


-- Populating the DONATION Table using INSERT statements
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DON
VALUES (7111, '20111', 'KIC Fridge', 5999, TO_DATE('2024-05-01', 'Y

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DON
VALUES (7112, '20112', 'Samsung 42inch LCD', 3999, TO_DATE('2024-05

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DON
VALUES (7113, '20113', 'Sharp Microwave', 1099, TO_DATE('2024-05-03
```

```sql
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DON
VALUES (7114, '20114', 'Glass Dining Room Table', 2999, TO_DATE('20

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DON
VALUES (7115, '20115', 'Lazyboy Sofa', 1199, TO_DATE('2024-05-07', 

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DON
VALUES (7116, '20113', 'Sound System', 179, TO_DATE('2024-05-09', '


-- Populating the DELIVERY Table using INSERT statements
INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, D
VALUES (511, 'Double packaging requested', TO_DATE('2024-05-10', 'Y

INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, D
VALUES (512, 'Delivery to work address', TO_DATE('2024-05-12', 'YYY

INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, D
VALUES (513, 'Signature required', TO_DATE('2024-05-12', 'YYYY-MM-D

INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, D
VALUES (514, 'No notes', TO_DATE('2024-05-12', 'YYYY-MM-DD'), TO_DA

INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, D
VALUES (515, 'Birthday present wrapping required', TO_DATE('2024-05

INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, D
VALUES (516, 'Delivery to work address', TO_DATE('2024-05-20', 'YYY


-- Populating the RETURNS Table using INSERT statements
INSERT INTO RETURNS (RETURN_ID, RETURN_DATE, REASON, CUSTOMER_ID, D
VALUES ('ret001', TO_DATE('2024-05-25', 'YYYY-MM-DD'), 'Customer no
VALUES ('ret002', TO_DATE('2024-05-25', 'YYYY-MM-DD'), 'Product hac


-- Populating the INVOICE Table using INSERT statements
INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOY
VALUES (8111, 11011, TO_DATE('2024-05-15', 'YYYY-MM-DD'), 'emp103',

INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOY
VALUES (8112, 11013, TO_DATE('2024-05-15', 'YYYY-MM-DD'), 'emp101',

INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOY
VALUES (8113, 11012, TO_DATE('2024-05-17', 'YYYY-MM-DD'), 'emp102',

INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOY
VALUES (8114, 11014, TO_DATE('2024-05-17', 'YYYY-MM-DD'), 'emp104',

INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOY
VALUES (8115, 11015, TO_DATE('2024-05-18', 'YYYY-MM-DD'), 'emp105',

INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOY
VALUES (8116, 11015, TO_DATE('2024-05-18', 'YYYY-MM-DD'), 'emp105',

-- SELECT statements to ensure values were properly implemented
SELECT * FROM CUSTOMER;
SELECT * FROM EMPLOYEE;
SELECT * FROM DONATOR;
SELECT * FROM DONATION;
SELECT * FROM DELIVERY;
SELECT * FROM RETURNS;
SELECT * FROM INVOICE;

-- Verification that relationships between tables are working (JO
SELECT INVOICE.INVOICE_NUM, CUSTOMER.FIRST_NAME, CUSTOMER.SURNAME,
FROM INVOICE
```

```
/*
Question 2
*/

/*
SQL Query to generate a report with combined customer name, emplo
delivery notes, donation purchased, invoice number, and invoice d
filtering for invoices after 16 May 2024.
*/

SELECT
    CUSTOMER.FIRST_NAME || ' ' || CUSTOMER.SURNAME AS CUSTOMER_NAME
    INVOICE.EMPLOYEE_ID,  -- Employee ID from the INVOICE table
    DELIVERY.DELIVERY_NOTES,  -- Delivery notes from the DELIVERY
    DONATION.DONATION,  -- Donation purchased from the DONATION ta
    INVOICE.INVOICE_NUM,  -- Invoice number from the INVOICE table
    TO_CHAR(INVOICE.INVOICE_DATE, 'DD/MON/YYYY') AS INVOICE_DATE  -
FROM
    INVOICE
JOIN
    CUSTOMER ON INVOICE.CUSTOMER_ID = CUSTOMER.CUSTOMER_ID  -- Joir
JOIN
    EMPLOYEE ON INVOICE.EMPLOYEE_ID = EMPLOYEE.EMPLOYEE_ID  -- Joir
JOIN
    DONATION ON INVOICE.DONATION_ID = DONATION.DONATION_ID  -- Joir
JOIN
    DELIVERY ON INVOICE.DELIVERY_ID = DELIVERY.DELIVERY_ID  -- Joir
WHERE
    INVOICE.INVOICE_DATE > TO_DATE('2024-05-16', 'YYYY-MM-DD');  --

/*
Question 3 (Completed in HRSchema.ADDB7311.A2.ST10263534)
*/
```

```sql
-- Set the session to use the specific container (in case of plug
ALTER SESSION SET CONTAINER = XEPDB1;

-- Create the HR user and set a password for the HR schema
CREATE USER hr IDENTIFIED BY hrpassword;

-- Grant the necessary privileges for the HR user to connect and
GRANT CONNECT, RESOURCE TO hr;

-- Provide a quota for the HR user to store data on the USERS tab
ALTER USER hr QUOTA UNLIMITED ON USERS;

-- Create a synonym for the Funding table in the HR schema, allow
CREATE SYNONYM funding FOR HR.Funding;

-- Select and display all rows from the Funding table to verify d
SELECT * FROM Funding;

-- Testing to see if a unique funding_id is generated for each ne
INSERT INTO Funding (funder, funding_amount)
VALUES ('Green Planet Initiative', 7000.00);

SELECT * FROM Funding;

/*
Solution Justification:

The requirement was to create a new table, `Funding`, with auto-g

1. **Table Creation:**
   I created the `Funding` table with three attributes: `funding_

2. **Sequence for Auto-Generation:**
   A sequence named `funding_seq` was created to ensure that each
```

```
3. **Trigger for Automation:**
   To automate the process of assigning a `funding_id` for each n
   The trigger ensures that before every new insertion, the seque
   This guarantees that users do not need to manually insert or t

4. **Workaround for SYS Limitations:**
   Initially, I encountered restrictions because the SYS schema,
   The SYS user is typically reserved for system-level operations

   To overcome this limitation:
   - I created a new user/schema, **HR**, with the necessary perm
   - I granted the SYS schema access to the `Funding` table and i
    This ensures that the SYS schema can interact with the data i

5. **Data Synchronization between SYS and HR:**
   To bridge the HR and SYS schemas, I created a **synonym** in t
   This approach allowed me to perform operations in the SYS sche
   This ensures that any updates in the HR schema, including auto

6. **Conclusion:**
   This solution guarantees that the `funding_id` is automaticall
   The HR schema acts as an intermediary, allowing the functional
*/

/*
Question 4
*/

SET SERVEROUTPUT ON;  -- Enabling server output to display the res

BEGIN
   -- A FOR loop to iterate through each record returned by the q
   FOR rec IN (
      SELECT
         c.FIRST_NAME || ', ' || c.SURNAME AS CUSTOMER,  -- Combine
         d.DONATION AS DONATION_PURCHASED,  -- Donation item purch
         d.PRICE,  -- Price of the donation
         r.REASON AS RETURN_REASON  -- Reason for the return
      FROM
         RETURNS r  -- The RETURNS table holds the return details
      JOIN
         CUSTOMER c ON r.CUSTOMER_ID = c.CUSTOMER_ID  -- Joining wi
      JOIN
         DONATION d ON r.DONATION_ID = d.DONATION_ID  -- Joining wi
   )
   LOOP
      -- Display the formatted output for each record
      DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || rec.CUSTOMER);  -- Displ
      DBMS_OUTPUT.PUT_LINE('DONATION PURCHASED: ' || rec.DONATION_P
      DBMS_OUTPUT.PUT_LINE('PRICE: ' || rec.PRICE);  -- Display the
      DBMS_OUTPUT.PUT_LINE('RETURN REASON: ' || rec.RETURN_REASON);
      DBMS_OUTPUT.PUT_LINE('---------------------------------');  --
   END LOOP;

END;
/

/*
Question 5
*/

SET SERVEROUTPUT ON;

BEGIN
   -- Fetch and display the customer's name, employee's name, don
   FOR rec IN (
      SELECT
```

```
            e.FIRST_NAME || ', ' || e.SURNAME AS EMPLOYEE,
            d.DONATION,
            dl.DISPATCH_DATE,
            dl.DELIVERY_DATE,
            dl.DELIVERY_DATE - dl.DISPATCH_DATE AS DAYS_TO_DELIVERY
        FROM
            CUSTOMER c
        JOIN
            INVOICE i ON c.CUSTOMER_ID = i.CUSTOMER_ID
        JOIN
            EMPLOYEE e ON i.EMPLOYEE_ID = e.EMPLOYEE_ID
        JOIN
            DONATION d ON i.DONATION_ID = d.DONATION_ID
        JOIN
            DELIVERY dl ON i.DELIVERY_ID = dl.DELIVERY_ID
    )
    LOOP
        -- Display the output directly from the cursor
        DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || rec.CUSTOMER);
        DBMS_OUTPUT.PUT_LINE('EMPLOYEE: ' || rec.EMPLOYEE);
        DBMS_OUTPUT.PUT_LINE('DONATION: ' || rec.DONATION);
        DBMS_OUTPUT.PUT_LINE('DISPATCH DATE: ' || TO_CHAR(rec.DISPATC
        DBMS_OUTPUT.PUT_LINE('DELIVERY DATE: ' || TO_CHAR(rec.DELIVER
        DBMS_OUTPUT.PUT_LINE('DAYS TO DELIVERY: ' || rec.DAYS_TO_DELI
        DBMS_OUTPUT.PUT_LINE('--------------------------------');
    END LOOP;

    -- Additional processing specifically for customer 11011
    FOR rec IN (
        SELECT
            dl.DISPATCH_DATE,
            dl.DELIVERY_DATE,
            dl.DELIVERY_DATE - dl.DISPATCH_DATE AS DAYS_TO_DELIVERY
        FROM
```

```
      JOIN
         DELIVERY dl ON i.DELIVERY_ID = dl.DELIVERY_ID
      WHERE
         i.CUSTOMER_ID = 11011
   )
   LOOP
      DBMS_OUTPUT.PUT_LINE('DAYS TO DELIVERY FOR CUSTOMER 11011: '
   END LOOP;

END;
/

/*
Question 6
*/

BEGIN
   -- Loop through each customer and calculate the total amount s
   FOR rec IN (
      SELECT
         c.FIRST_NAME,
         c.SURNAME,
         SUM(d.PRICE) AS TOTAL_AMOUNT
      FROM
         CUSTOMER c
      JOIN
         INVOICE i ON c.CUSTOMER_ID = i.CUSTOMER_ID  -- Join with I
      JOIN
         DONATION d ON i.DONATION_ID = d.DONATION_ID  -- Join with
      GROUP BY
         c.FIRST_NAME, c.SURNAME
   )
   LOOP
      -- Display the customer name and total amount
      DBMS_OUTPUT.PUT_LINE('SURNAME: ' || rec.SURNAME);
      DBMS_OUTPUT.PUT_LINE('AMOUNT: R ' || rec.TOTAL_AMOUNT);

      -- Determine the rating based on total amount
      IF rec.TOTAL_AMOUNT >= 1500 THEN
         DBMS_OUTPUT.PUT_LINE('RATING: (***)');
      END IF;

      -- Print separator
      DBMS_OUTPUT.PUT_LINE('---------------------------------');
   END LOOP;
END;
/

/*
Question 7
*/

-- Question 7.1
DECLARE
   -- Declare a variable to store the price of a donation using %
   v_price DONATION.PRICE%TYPE;

BEGIN
   -- Select the price of a specific donation and assign it to v_
   SELECT PRICE INTO v_price
   FROM DONATION
   WHERE DONATION_ID = 7111;

   -- Output the result
   DBMS_OUTPUT.PUT_LINE('The price of the donation is: R '|| v_pri
END;
/
```

```
DECLARE
   -- Declare a record variable to store an entire row from the C
   v_customer CUSTOMER%ROWTYPE;

BEGIN
   -- Fetch the customer data into v_customer
   SELECT * INTO v_customer
   FROM CUSTOMER
   WHERE CUSTOMER_ID = 11011;

   -- Output the customer's details
   DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer.FIRST_NAME
   DBMS_OUTPUT.PUT_LINE('Address: ' || v_customer.ADDRESS);
   DBMS_OUTPUT.PUT_LINE('Contact: ' || v_customer.CONTACT_NUMBER);
END;
/

--Question 7.3
DECLARE
   -- Declare a user-defined exception
   e_no_customer EXCEPTION;

   -- Declare a variable to store the customer name
   v_customer_name CUSTOMER.FIRST_NAME%TYPE;

BEGIN
   -- Attempt to fetch a customer name
   SELECT FIRST_NAME INTO v_customer_name
   FROM CUSTOMER
   WHERE CUSTOMER_ID = 99999;  -- Non-existent customer ID to trig

   -- Output the result if found
   DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_name);
   -- Handle the built-in NO_DATA_FOUND exception
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Error: Customer does not exist.');
END;
/

/*
Question 8
*/

SELECT
    c.FIRST_NAME,
    c.SURNAME,
    SUM(d.PRICE) AS AMOUNT,
    CASE
        WHEN SUM(d.PRICE) >= 1500 THEN '***'
        WHEN SUM(d.PRICE) BETWEEN 1000 AND 1499 THEN '**'
        ELSE '*'
    END AS CUSTOMER_RATING
FROM
    CUSTOMER c
JOIN
    INVOICE i ON c.CUSTOMER_ID = i.CUSTOMER_ID
JOIN
    DONATION d ON i.DONATION_ID = d.DONATION_ID  -- Correctly join
GROUP BY
    c.FIRST_NAME, c.SURNAME;
```